200208565-1

METHOD AND APPARATUS FOR PROVIDING SECURITY FOR A COMPUTER SYSTEM

BY:

MICHAEL F. ANGELO
3303 AMBER FOREST DRIVE
HOUSTON, TEXAS 77068

E. DAVID NEUFELD
19900 INDIGO LAKE DRIVE
MAGNOLIA, TEXAS 77355

DAVID HEISEY
9611 EAST SAVILE CIRCLE
HOUSTON, TEXAS 77065

METHOD AND APPARATUS FOR PROVIDING SECURITY FOR A COMPUTER SYSTEM

BACKGROUND OF THE INVENTION

[0001] This section is intended to introduce the reader to various aspects of art, which may be helpful in providing the reader with background information to facilitate a better understanding of the various aspects of the present techniques. Accordingly, it should be understood that these statements are to be read in this light, and not as admissions of prior art.

[0002] In the field of processor-based computer systems, it may be desirable for information to be transferred from a computer system to another computer system via a network. Computer networks may be arranged to allow information, such as files or programs, to be shared across an office or any geographic boundary. As an aspect of efficiently maintaining the exchange of information, computer systems in a network may include various security systems, such as programs or devices, to prevent unauthorized intrusions or attacks from outside sources. These security systems, for example, may prevent malicious or unknown code that corrupt data and programs stored on the computer system.

[0003] In providing security for a computer system, computer systems typically preclassify viruses to assist in identifying malicious code. However, when new viruses are introduced, computer systems are vulnerable because a virus may be unknown or unclassified. As a result, the computer system is not able to remove an unknown virus before it attacks the computer system. In addition, the performance of the central processing unit ("CPU") may be impacted by the operation of security functions of the computer system. The computer system's overall performance may be diminished because the security functions are consuming the resources of the CPU.

SUMMARY OF THE INVENTION

[0004] The present invention relates generally to a technique of providing security for a computer system. In the technique, a request may be generated for a file. The dedicated security processor receives the request for the file and accesses the requested file. Then, the dedicated security processor validates the requested file and provides the file to another processor, if the requested file is validated.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Exemplary embodiments of the invention are apparent upon reading the following detailed description with reference to the drawings in which:

[0006] FIG. 1 is a block diagram illustrating a computer network in accordance with embodiments of the present invention;

[0007] FIG. 2 is a block diagram illustrating a computer system in a network in accordance with embodiments of the present invention;

[0008] FIG. 3 is a process flow diagram illustrating a security process in accordance with embodiments of the present invention;

[0009] FIG. 4 is a process flow diagram illustrating a security process that verifies the identity of a requestor in accordance with embodiments of the present invention; and

[0010] FIG. 5 is a process flow diagram illustrating a security process that verifies an identifying number associated with a request in accordance with embodiments of the present invention.

DESCRIPTION OF SPECIFIC EMBODIMENTS

[0011] One or more specific embodiments of the present invention will be described below. In an effort to provide a concise description of these embodiments, not all features of an actual implementation are described in the specification. It should be appreciated that in the development of any such actual implementation, as in any engineering or design project, numerous implementation-specific decisions may be made to achieve the developers' specific goals, such as compliance with system-related and business-related constraints, which may vary from one implementation to another. Moreover, it should be appreciated that such a development effort might be complex and time consuming, but would nevertheless be a routine undertaking of design, fabrication, and manufacture for those of ordinary skill having the benefit of this disclosure.

[0012] The disclosed embodiments provide an improved approach that may address one or more of the issues discussed above, while enhancing the performance of a computer system. With computer systems, security functions may be provided to protect the system. The security functions may be managed by a device or component, such as a processor, that is within the computer system or external to the computer system. In the disclosed embodiments, the security of the computer system is maintained in a manner that: (1) protects against defeat by thread models or technologies; (2) minimizes interaction with the CPU; and (3) allows trapping of code that is unknown or unclassified.

[0013] For instance, while not limited in any way to such applications, the disclosed embodiments may enable a computer system to operate in a more efficient manner by having a security co-processor that protects against defeat by thread models or technologies. Threaded programs allow background and foreground action to take place without the overhead of launching multiple processes or inter-process communication. By having a security co-processor examine code and activities that are independent of the operating system, the threaded programs are unable to defeat the security of the computer system. In addition, the security co-processor may minimize the performance impact on the central processing unit ("CPU") of a computer system by performing the security functions, which allows the CPU to devote more resources to non-security related functions. Furthermore, the security co-processor may examine new code without the code being pre-classified. Thus, as new viruses are introduced, the security co-processor may trap the unknown or unclassified code before the CPU is damaged by an attack from the code.

[0014] Referring initially to FIG. 1, a block diagram of a computer network architecture is illustrated and designated using a reference numeral 10. A server 20 may be connected to a plurality of client computers 22, 24 and 26. The server 20 may be connected to as many as "n" different client computers. Each client computer in the network 10 may be a functional client computer and may be a desktop personal computer ("PC"), a notebook PC, a tablet PC, a personal digital assistant ("PDA"), for example. The magnitude of "n" may be a function of the computing power of the server 20.

[0015] The server 20 may be connected via a network infrastructure 30, which may include any combination of hubs, switches, routers, etc. While the network infrastructure 30 is illustrated as being either a local area network ("LAN"), storage area network ("SAN"), a wide area network ("WAN"), or a metropolitan area network ("MAN"), those skilled in the art will appreciate that the network infrastructure 30 may assume other forms or may even provide network connectivity through the Internet. As described below, the network 10 may include other servers as well, which may be dispersed geographically with respect to each other to support client computers in other locations.

[0016] The network infrastructure 30 may connect the server 20 to the server 40, which may be representative of any other server in the network environment. The server 40 may be connected to one or more client computers 42, 44, and 46. As illustrated in FIG. 1, a network infrastructure 50, which may include a LAN, a WAN, a MAN, or other network configuration, may be used to connect the client computers 42, 44 and 46 to the server 40. The server 40 may additionally be connected to the Internet 60, which may be connected to a server 70. The server 70 also may be connected to one or more of client computers 72, 74 and 76.

[0017] In the computer network 10, a wide array of problems may occur. For example, a virus or malicious code may attack the server 40, which may result in other systems, such as the plurality of client computers 42, 44, and 46, being impacted. Also, the server 40 may have to respond to various requests and code that may diminish performance. If the server 40 supports a large number of clients, such as a plurality of client computers 42, 44, and 46, a large customer or client base may experiences problems, which may result in delays in operation. In addition, if the server 40 is attacked by an unknown or unclassified virus, then the virus may damage information within the server 40 or result in downtime for the computer network 10.

[0018] FIG. 2 is a block diagram illustrating an exemplary computer system in accordance with embodiments of the present invention. The computer system, which may include client computers 22, 24, 26, 42, 44, 46, 72, 74, 76 or servers 20, 40, 70, as discussed above, is generally referred to by the reference numeral 100. The architecture of the computer system 100 is given for purposes of illustration only, as computer systems in which the present teachings are applicable may include additional components or a subset of the components illustrated in FIG. 2.

[0019] The computer system 100 may comprise a processor complex 102, which may include a plurality of central processing units ("CPUs"). A core logic chipset 104, which may manage a variety of functions on behalf of the processor complex 102, may be connected to the processor complex via a processor bus 103.

[0020] The core logic chipset 104 may be connected via memory bus 105 to a system random access memory, which may comprise static random access memory ("SRAM"), dynamic random access memory ("DRAM") or other suitable memories. The memory may be a shared system memory to hold memory resident files. A video graphics controller 110 may be connected to the core logic chipset 104 via a video bus 107 to provide a signal that produces a display image on a video display 112.

[0021] A bus 109 such as a peripheral component interface ("PCI") bus or the like may connect the core logic chipset to a variety of system devices, such as a network interface card 122 and a PCI/PCI bridge 124. The network interface card 122 may provide communication capability to the computer system 100 via a communication bus 119. The communication bus

119 may be connected to other computer systems, as discussed above. The PCI/PCI bridge 124 may provide capacity for additional PCI devices on a PCI bus 117.

[0022] A PCI/SCSI bus adapter 114 may provide access to SCSI devices such as a disk drive 130 and a tape drive 132 via a SCSI bus 131. A PCI/ATA controller 118 may provide access to additional devices such as a disk drive 128 and a CD ROM drive 134. A PCI/EISA/LPC bus may provide access to system devices such as a read only memory basic input/output system ("ROM BIOS") 140, a non-volatile memory 142 (such as flash memory or the like), a modem 120 or the like via a bus 113. The modem may provide communication access via a phone line 121. An input/output controller 126, which may be connected to the bus 113 may provide access to system devices such as a floppy disk drive 150, a serial port 152 a real time clock (" RTC") 154, a CD ROM drive 144, a keyboard 146, a mouse 148 and the like.

[0023] A security co-processor 111 may be connected to the bus 109 to perform security functions as more fully described below. The security co-processor 111 may be configured to validate certain functions and activities before they are actually executed. Also, the security co-processor 111 may be dedicated to providing security functions to the computer system 100. Because the security co-processor 111 performs the security functions and activities, it frees the use of the CPU cycles on the processor complex 102 for other computing activities.

[0024] The security co-processor 111 may examine code independent of the operating system ("OS") executing in the computer system 100 or other computers in the computer network 10. For example, the security co-processor 111 may enable the computer system 100 to prevent thread technologies and unknown code from attacking the computer system 100.

As a benefit to the computer system 100, the security co-processor 111 may examine code independently of the processor complex 102, which may be executing an operating system. As such, the security co-processor 111 may trap code that is unknown or unclassified to prevent it from impacting the performance or integrity of the computer system 100. Beneficially, the security co-processor 111 does not have to access stored information of known viruses or attack signatures to be able to prevent malicious code.

[0025] To communicate with the processor complex 102 or other components within the computer system 100, the security co-processor 111 may include indications regarding the file or code being accessed. For instance, the security co-processor 111 may include a return status field, such as a semaphore, which may be a flag to indicate whether the file or resource is being used by another thread or process. The return status field may be within a memory or a register, to name a few examples. The return status field may be used by the security co-processor 111 to communicate status information about the files that it validates.

[0026] As noted above, it may be beneficial to protect the computer system 100 from unauthorized access by external systems, users, or programs. For instance, providing desired security may include protecting a system from viruses or attacks by hackers by implementing a security co-processor 111. One way to enhance security in the computer system 100 along with the security co-processor 111 may be to prevent or restrict access to system passwords without authentication that the access is by an authorized user. FIG. 3 shows an example of how the security co-processor 111 may operate to prevent unauthorized access to files stored on the computer system 100.

[0027] FIG. 3 is a process flow diagram illustrating a security process in accordance with embodiments of the present invention. In the process flow diagram 200, which may be best understood by concurrently viewing FIG. 2, a system, such as the computer system 100 of FIG. 2, may include a processor and a security co-processor. The processor may correspond to the processor complex 102 of FIG. 2 or another computer, and the security co-processor may correspond to the security co-processor 111 of FIG. 2. The process begins at block 202. The process involves the validation of a file in response to a command or request from a processor that may be an internal processor 102 to the computer system 100 or an external processor at a remote system. The security co-processor 111 may validate the request prior to the execution of certain commands on the system, while operating independently of the OS on the system. At block 204, the processor 102 may generate a request, which may be a request for a file. The request may include the filename, path, error correction information, identifying data, user identification, system identification, a password or other suitable information. In addition, the information or portions of the information within the request may be hashed or signed with a key. The processor 102 may transmit the request to a component, such as an I/O driver, which may be a part of the core logic chipset 104 or be a software program, which forwards the request to the security co-processor 111 in block 206.

[0028] At block 208, the security co-processor 111 may locate the file that corresponds to the request. The file may be located within the system memory 106 or 146. Then, at block 210, the security co-processor 111 may access or look up the file. In accessing the file, the security co-processor may access a record that corresponds to the requested file. The record may be a signed or hashed version of information that corresponds to users or systems authorized to access the file. The record may include similar information to the request, such as a filename, path, error correction information, identifying data, user identification, system identification, a password or other suitable information. Also, the record may be stored within a memory 106 or 146, a device, a local database, or a remote database that is accessible by the security co-processor 111. If the record is stored in a database, the security co-processor 111 may communicate with the database through a protocol, such as Lightweight Directory Access Protocol ("LDAP"). LDAP is a directory service protocol that may operate over TCP/IP. The record may include a signature or hash for verifying the authenticity or integrity of the requested file, such as a digital signature. This may allow the system 100 to determine if the file has been altered by signing a portion of the file, a compressed version of the file, or the complete file. Similarly, the system 100 may preload the file, as a memory resident file, into a cache or other similar memory location, such as system memory 106 or 146.

[0029] Digital "signatures" may be used to provide authentication of the file. A digital signature is a data component, such as a hash, that may have been encrypted using a private key authentication process or the like. The encryption process creates a unique signature, which may allow verification of a data file. The use of digital signatures may allow the security co-processor 111 to validate that the file requested is from a trusted source and/or that the file has not been subsequently altered by anyone else.

[0030] Additionally, the record associated with the request may include validation data such as an error checking and correction ("ECC") code. The ECC code may be implemented to repair potential problems with the file. For instance, if a file is corrupted or other problems exist with the file, the ECC code may be used to correct the file back to its original form. This may allow the system 100 to operate more efficiently because files may be corrected without involving the processor 102 or other failure routines.

[0031] At block 212, the security co-processor 111 may determine if the requested file is valid. The requested file may be verified to determine if the requested file is valid to perform certain commands or operations or if the requested file has been altered. The verification of the requested file may involve accessing information within the request, from a database, and/or memory location and comparing it with the file, a portion of the file, or the record, which may include information as discussed above. For example, the validation of the requested file may involve verifying a digital signature in the request against data within the record to authenticate or verify the integrity of the file. Further, the validation may involve verifying the ECC code of the file against data within the record. As another example, the request may be transmitted over LDAP, which may include information that verifies the authorization to access the files.

[0032] If the requested file is not valid, then a return status field may be set in block 214 to indicate that the file return is invalid. The setting may create a status message that indicates that the request is invalid, set a return status field to invalid, or simply abort the access. Advantageously, this allows the security co-processor to trap a file that fails authenticity or integrity verification checks against the record, validated version, or signed file. This may prevent new system or code attacks without pre-classification. However, if the requested file is valid, then the request may be executed at block 216. The execution of the request may involve delivering the file to the security co-processor 111, executing the command in the request, or other similar function. After the file is delivered to the security co-processor 111, a status return field may be set in block 218 to indicate that the file return is valid. The status return field setting may create a status message that indicates that the request is valid, set a return status to "return file good," or simply indicate that the file is access is complete.

[0033] After the return status field is set for the request, the security co-processor 111 may set the return status to "good" for the security validation for the request at block 220. The setting of the return status to "good" may result in a message being sent to the processor or may allow the security co-processor 111 to free up resources for other requests or may involve sending the file to the processor 102 for processing. Once the return status field is set for the request, the process may continue to process the file without the involvement of the security co-processor 111 at block 222. Thus, the process ends at block 224.

[0034] To provide additional security to the computer system 100, the requests to the security co-processor 111 may include the identity of a requestor. By including the identity of the requestor in the request, the security co-processor 111 may be able to provide added protection for certain files or prevent unauthorized access to certain information. A process flow illustrating a security process that verifies the identity of a requestor in accordance with embodiments of the present invention is shown in FIG. 4.

[0035] FIG. 4 is a process flow diagram, which may be best understood by concurrently viewing FIG. 2, illustrating the use of a security co-processor 111 to process requests and to verify the identity of a requestor in accordance with embodiments of the present invention. In this process flow diagram 300, a system, such as the computer system 100 of FIG. 2, may include a processor 102 and a security co-processor 111. The process begins at block 302. In addition to the validation of requested file, the process may involve the validation of the user accessing the file. The security co-processor 111 may validate the requestor prior to the operation or execution of certain commands by the system. At block 304, the processor 102 may generate a request, which may be a request for a file or command. The processor 102 may transmit the request to a component, such as an I/O driver, that forwards the request to the security co-processor 111 in block 306, similar to step 206 of FIG. 3 discussed above.

[0036] At block 308, the security co-processor 111 may locate the file that corresponds with the request. Then, at block 310, the security co-processor 111 may access the file. The access of the file may be similar to the file lookup in block 210 of FIG. 3, which may include a record having a digital signature to verify authenticity or the integrity of the file. As previously discussed, the record may be stored within a variety of locations, such as a memory 106 or 146 that is coupled to the security co-processor 111, and may communicate with the database through a protocol, such as LDAP.

[0037] At block 312, the security co-processor 111 may determine if the requested file is valid. The requested file may be verified as discussed above with regard to block 212 of FIG. 3. If the requested file is not valid, then a return status field may be set in block 314 to indicate that the file return is invalid. The setting of the return status field may create a status message that indicates that the request is invalid, set a return status field to invalid, or simply abort the access. Again, this allows the security co-processor 111 to trap a file that fails to be verified, as discussed above. However, if the requested file is valid, then the request may be executed at block 316. The execution of the request may involve delivering the file to the security co-processor 111, executing the command in the request, or other similar function. After the file is transmitted to the security co-processor 111, a status return field may be set in block 318 to indicate that the file return is valid. The status return field setting may create a status message that indicates that the request is valid, set a return status to "return file good," or simply indicate that the file is access is complete.

[0038] At block 320, the security co-processor 111 may validate whether the user who requested access to the file or command is an authorized user. In validating the user access, the security co-processor 111 may access a record that corresponds to the user access being requested. The record may include a signature for verifying the authenticity of the user requesting the file or the execution of a command. Similar to the file discussed above, the authenticity of the user access may be provided by a digital signature. The digital signature may be created through any manner, such as a private and public key or hash algorithm, for example. The user access may include commands, authorized files or other information relating to the user. At block 322, the security co-processor 111 may determine if the user access is valid. The verification of the user access may be through accessing information within the security co-processor 111, from a database, or memory location. The information may then be matched against the user access information to determine if the user access is valid.

[0039] If the user access is valid, then the return status field may be set to "valid" at block 322. However, if the user access is not valid, then a return status field may be set in block 314 to indicate that the return status is invalid. As discussed before, the return status field being set to "invalid" may result in a message being generated or the processor 102 being notified of the failed request. After the return status field is set for the user access, the security co-processor 111 may set the return status field to "good" for the security validation at block 324. The setting of the return status field to "good" may result in a message being sent to the processor 102, may allow the security co-processor 111 to free up resources for other requests, or may involve sending the file to the processor 102 for processing. Once the return status field is set for the request, the process may continue to process the file without the involvement of the security co-processor 111 at block 326. The process ends at block 328.

[0040] In an alternative embodiment, the lookup of the user access of block 320 may be implemented after the security co-processor 111 looks up the file in block 310. Then, the user access verification of block 322 may be performed. After the user access has been verified, the validation of requested file may take place at block 312. Accordingly, the verification process may be further variety for other designs.

[0041] To further enhance the security, the requests to the security co-processor 111 may include a number or value to identify a requestor. The identifying value may be a nonce, which is a randomly generated number, or may be a time stamp that is associated with the time of the request. By including the identifying value, the security co-processor 111 may be able to provide added protection for certain files or prevent unauthorized access to certain information. In addition, the identifying value may be exchanged before the request is received by the security co-processor 111, delivered after the request has been sent, or as part of the requesting process. As a result, the security co-processor may prevent playback attacks or other similar attacks from being able to access the computer system. Using the identifying value may provide the computer system 100 with increased security without burdening the processor 102. A process flow illustrating a security process that verifies the number or value identifying a requestor in accordance with embodiments of the present invention is shown in FIG. 5.

[0042] FIG. 5 is a process flow diagram, which may be best understood by concurrently viewing FIG. 2, illustrating the use of a security co-processor 111 to process requests and verify a nonce, which may be a random number to identify a requestor, in accordance with embodiments of the present invention. In this process flow diagram 400, a system, such as the computer system 100 of FIG. 2, may include a processor 102 and a security co-processor 111. The process begins at block 402. In addition to the validation of a command or request in FIG. 3, the process involves the use of an identifying number to validate the requestor of the command or file. The security co-processor 111 may validate the requestor prior to the operation of certain commands on the system. At block 404, the processor 102 may generate a request along with an identifying value, which may be a request for a file or command. The request may include a variety of information, such as a filename, path, user identification, or system identification, as discussed above. In addition, the request may also include hashed or signed information that may be associated with a time stamp, a random number, or nonce as well. The identifying value or number may be a nonce or time stamp, as discussed above, which may be exchanged before the request is received by the security co-processor 111. The processor 102 may transmit the request along with the identifying number to a component, such as an I/O driver, that forwards the request to the security co-processor 111 in block 406, which is similar to step 206 of FIG. 3 discussed above.

[0043] At block 408, the security co-processor 111 may locate the file that corresponds with the request. Then, at block 410, the security co-processor 111 may access the file. The access of the file may be similar to the file lookup in block 210 of FIG. 3, which may include a record having a digital signature for authenticity, integrity, or error correction code ("ECC"). The record may also include hashed or signed information that may be associated with a time stamp, a random number, or nonce. As previously discussed in FIG. 3, the record may be stored within a memory that is coupled to the security co-processor 111 and may communicate with the database through a protocol, such as LDAP.

[0044] At block 412, the security co-processor 111 may determine if the requested file is valid. The requested file may be verified as discussed above with regard to block 212 of FIG. 3. If the requested file is not valid, then a return status field may be set in block 414 to indicate that the file return is invalid, as discussed above with regard to block 214 of FIG. 3. As noted above, the security co-processor 111 may trap a file that fails to be verified without pre-classification. However, if the requested file is valid, then the request may be executed at block 416, as discussed above with regard to block 216 of FIG. 3. After the file is transmitted to the security co-processor 111, a status return field may be set in block 418 to indicate that the file return is valid.

[0045] At block 420, the security co-processor 111 may look up the identifying number. In looking up the identifying number, the security co-processor 111 may access a record that corresponds to the identifying number within the request. The record may include an identifying number, such as a randomly generated number for the communication that may be used to provide additional security. This protection may prevent playback attacks from circumventing security of the computer system 100. The authenticity of the identifying number may be hashed or signed to provide additional protection as discussed above with regard to the file or request. At block 422, the security co-processor 111 may determine if the identifying number is valid. The verification of the identifying number may be through accessing information within the security co-processor 111, from a database, or memory location. The information may then be matched against the identifying number to determine if the user is authorized or the command is valid.

[0046] If the identifying number is valid, then the return status field may be set to "valid" at block 422. However, if the identifying number is not valid, then a return status field may be set in block 414 to indicate that the return status is invalid. As discussed before, the return status field being set to "invalid" may result in a message being generated or the processor 102 being notified of the failed request. After the return status field is set for the user access, the security co-processor 111 may set the return status field to "good" for the security validation at block 424. The setting of the return status field to "good" may result in a message being sent to the processor 102, may allow the security co-processor 111 to free up resources for other requests, or may involve sending the file to the processor 102 for processing. Once the return status field is set for the request, the process may continue to process the file without the involvement of the security co-processor 111 at block 426. The process ends at block 428.

[0047] Alternatively, the lookup of the identifying number, such as the nonce or time stamp, in block 420 may be implemented after the security co-processor 111 looks up the file in block 410. Then, the identifying number verification of block 422 may be performed. After the identifying number has been verified, the validity of the requested file may be determined at block 412. As another alternative embodiment, the verification of user access 320 and 322 of FIG. 4 may be implemented in the process flow diagram 400 of FIG. 5 after block 410 and before block 426. Accordingly, the verification process may be further variety for other designs.

[0048] While the invention may be applicable to various modifications and alternative forms, specific embodiments have been shown by way of example in the drawings and have been described in detail herein. However, it should be understood that the invention is not intended to be limited to the particular forms disclosed. Rather, the invention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the invention as defined by the following appended claims.